Adjutant: an R-based tool to support topic discovery for systematic and literature reviews

Anamaria Crisan, Tamara Munzner, Jennifer L. Gardy

Adjutant is primarily a graphical user interface that implements a standard text mining workflow in addition to t-SNE and hdbscan under the hood to rapidly analyze a corpus of PubMed articles and to derive topic clusters in an unsupervised manner. This document details Adjutant’s implementation and then investigates the quality of Adjutant’s clustering abilities using both synthetic and real data.

Please note that this document is written in the conversational style of a tutorial and is available as an R markdown notebook at: https://github.com/amcrisan/adjutant/#important-adjutant-details

Contents

1.0 ADJUTANT IMPLEMENTATION DETAILS 2

2.0 EVALUATION 3

2.1 OVERALL EVALUATIVE APPROACH 3

2.2 EVALUATION WITH SIMULATED DATA 4

2.2.1 A TWO CLASS EXAMPLE 4
2.2.1.1 Running t-SNE 5
2.2.1.2 Running Adjutant’s hdbscan procedure 8

2.2.2 A MULTI-CLASS EXAMPLE 11
2.2.2.1 Generating sample data 11
2.2.2.2 Running t-SNE 12
2.2.2.3 Running Adjutant’s hdbscan Procedure 13
2.2.2.4 Adding Noise 16

2.3 EVALUATION WITH REAL DATA 18

2.3.1 OBTAINING REAL-WORLD DATA AND RUNNING ADJUTANT 18

2.3.2 VALIDITY OF CLUSTERS 22
2.3.2.1 Visualizing the distribution of specific terms 22
2.3.2.2 Visualizing cluster similarity 24

3.0 COMPARISON TO LDA 27
1.0 Adjutant Implementation Details

This section provides low-level details of Adjutant’s algorithmic implementation.

**Querying PubMed and assembling a document corpus.** Given a PubMed-compatible search query, Adjutant uses the RISmed package (Kovalchik, 2016) to obtain information about articles, including PubMed ID, Journal, Article Title, Authors, Abstract, Public Date, and MeSH terms. It then uses the jsonlite package (Ooms, 2014) and the E-Utils eSummary API to query and extract additional metadata, including PubMed Central (PMC) ID, article DOI, PMC citation count, article type (i.e. Journal Article, Review, Meta-Analysis), and language. Both RISmed and jsonlite are used because of the differing outputs from the E-Utils eFetch and eSummary APIs.

**Data wrangling.** Adjutant follows a relatively standard text mining workflow laid out in the tidytext manual (Silge and Robinson, 2016). The process begins with Adjutant decomposing the PubMed document corpus into single-word entities extracted from article titles and abstracts and converts this text to a tidy format for further analysis using the tidytext package (Silge and Robinson, 2016). All words are stemmed using Porter's algorithm (Van Rijsbergen et al., 1980), from the SnowballC package (Bouchet-Valat, 2014), after which common (stemmed) stop words are removed. Adjutant next calculates the term frequency inverse document frequency (tf-idf) metric, again using tidytext package resource, and then filters terms that are too infrequent (fewer than 1% of all documents) or too frequent (more than 70%). Finally, Adjutant generates a document term matrix (DTM), with articles as rows, stemmed single words as columns, and tf-idf as the relevant analytic metric.

**Unsupervised Topic Clustering.** The multidimensional DTM is mapped into two dimensions using the Barnes-Hut t-SNE (van der Maaten, 2014) implementation from the Rtsne package (Krijthe, 2015). We use default t-SNE parameters, except when the document corpus contains more than 1000 articles and when the t-SNE perplexity parameter (Wattenberg et al., 2016) is set to 50; however, Adjutant also allows users to modify the perplexity and theta t-SNE parameters after an initial analysis is complete. Next, Adjutant derives clusters using the hdbscan algorithm (Campello et al., 2013) from the dbscan package (Hahsler and Piekenbrock, 2017). Adjutant will attempt to automatically calculate a good hdbscan minimum cluster size (minPts) parameter, with ‘good’ being defined here as the fewest number of clusters that best fits the dimensionally reduced data. Adjutant identifies a good minPts parameter value by leveraging goodness-of-fit measurements derived from linear models, specifically the adjusted R² and the Bayesian Information Criteria (BIC); thus, each minPts parameter value tested will have an associated R² and BIC measure. Adjutant makes this calculation by fitting separate linear models to each of the two t-SNE dimensions, where for each linear model the t-SNE component coordinates are used as the dependent variable and the clusters are used as the independent variables. Each cluster is a vector of membership probabilities, from 0 (not in the cluster) to 1 (definitely a cluster member). The adjusted R² between the two component models are multiplied, and the BICs are averaged. To choose the optimal minPts parameters, Adjutant identifies all minPts values with an adjusted R² within 0.05 of the best performing minPts value, and among those different options selects the minPts value with the lowest BIC. The clusters resulting from the optimal minPts value are named using the two most commonly occurring terms within the cluster.

In the analysis that follows, we examine the performance of Adjutant’s unsupervised topic clustering methods with both synthetic and real data.
Implementation References


2.0 Evaluation

2.1 Overall Evaluative Approach

This section provides an overview of Adjutant’s analytic workflow using both real and synthetic data. We include both the results and associated R code and remind the reader that they can run this analysis for themselves through the markdown notebook available online at:

https://github.com/amcrisan/adjutant/#important-adjutant-details

Ahead of the evaluation details, we would like to provide an overview of our evaluative approach and a summary of our findings. We begin with an analysis of a two cluster synthetic dataset (Section 2.2.1) and then extend that analysis to more complex multi-cluster synthetic data with the addition of random noise (Section 2.2.2). We found that in all of these conditions Adjutant is capable of recovering the initial cluster structure entirely or to a reasonable extent when noise is introduced. Next, we used a real dataset to assess the validity of Adjutant’s clustering (Section 2.3). Absent a ground truth data, we extensively compared how terms from the documents are spread across the resolved clusters and demonstrate that both the t-SNE co-ordinate space and the cluster boundaries yielded reasonable results (Section 2.3.2). We demonstrated this result by performing a cosine similarity of clusters and showing that documents within a cluster are more similar to each other relative to those in adjacent and also more distal clusters. Relevant ground truth data for the purposes of conducting systematic reviews is difficult to obtain and in fact variable depending on the goals of the individual conducting the analysis. By demonstrating that our cluster results and the spatial arrangement of documents are reasonable we are
intending to demonstrate that algorithm works well, but we still expect a user to interpret Adjutant’s findings. Finally, we compare Adjutant to a published and established LDA analysis workflow to demonstrate that LDA struggles to resolve sensible clusters in the same real-world dataset example (Section 3.0).

To run the evaluation analysis that follows we recommend that the following packages be installed.

```r
library(MASS)
library(ggplot2)
library(adjutant)
library(dplyr)
library(Rtsne)
library(dbscan)
library(tidytext)
library(reshape)
library(ggthemes)
library(cowplot)
library(topicmodels)
```

### 2.2 Evaluation with Simulated Data

Prior to testing Adjutant with a real data set, we will explore Adjutant’s abilities with some generated data, thus knowing the ground truth. Note that the Distill Pub article on t-SNE (Wattenberg et al., 2016) does a similar analysis, but in this supplemental materials we are testing the specific configuration and environmental dependencies upon which Adjutant is built.

#### 2.2.1 A two class example

The Wikipedia article for t-SNE advises against clustering on t-SNE dimensionally reduced data points and references a Stats Exchange discussion ([https://stats.stackexchange.com/questions/263539/k-means-clustering-on-the-output-of-t-sne/264647#264647](https://stats.stackexchange.com/questions/263539/k-means-clustering-on-the-output-of-t-sne/264647)). In Adjutant’s implementation we used t-SNE ahead of hbdscan because it resulted in a significant speed-up and without a substantial cost in accuracy (which we will demonstrate in these supplemental materials). Absent in the Stats Exchange discussion is a compelling demonstration of how well or how poorly a clustering algorithm would perform. Here we begin by assessing the validity of the claims made within the Stats Exchange discussion.

First, we will generate roughly the same two distributions presented in Stats Exchange discussion:

```r
set.seed(416)

multiNormGen<-function(mu = rep(0,2),Sigma=matrix(c(10,3,3,2),2,2),grpName = NULL,n=1000){
  values<-mvrnorm(n = n, mu = mu, Sigma = Sigma)
  if(!is.null(grpname)){
    return(data.frame(x=values[,1],y=values[,2],grp=rep(grpname,nrow(values))))
  }else{
    return(data.frame(x=values[,1],y=values[,2]))
  }
}
sampleDat<-rbind(multiNormGen(n=250,mu=c(-2,0),Sigma = matrix(c(1,0,0,1),2,2),grpName = "group1"),
                multiNormGen(n=750,mu=c(2,0),Sigma = matrix(c(1,0,0,1),2,2),grpName = "group2"))
```
2.2.1.1 Running t-SNE

We will now run t-SNE on these two derived distributions. Since Adjutant expects a Document Term Matrix (DTM) as input, we have instead supplied Adjutant’s logic for choosing a perplexity parameter for t-SNE:

```r
# first, adjutant selects a perplexity depending upon the number of articles
tsnePer <- 30 # default adjutant value
if(nrow(sampleDat) >= 1000){
  tsnePer <- 50
} else if(nrow(values$corpus) <= 100){
  tsnePer <- 5
}

# then it runs t-SNE
tsneObj <- Rtsne(sampleDat[, c("x", "y")], perplexity = tsnePer)

# some cleaning up and renaming to satisfy the next steps
df <- data.frame(cbind(1:nrow(tsneObj$Y), tsneObj$Y), stringsAsFactors = F)
colnames(df) <- c("PMID", paste("tsneComp", 1:(ncol(tsneObj$Y)), sep=""))

# let's take a look at the t-SNE output
df <- inner_join(sampleDat, df)

ggplot(df, aes(x = tsneComp1, y = tsneComp2, color = grp)) +
  geom_point(alpha = 0.7) +
  theme_bw()
```
In the above figure it is possible to see that the larger cluster is spread out quite a bit so as to make it look like there are many other smaller clusters within it. This observation concurs with the Stats Exchange discussion. These visibly smaller clusters have the potential to impact hdbscan results. Prior to moving on to the clustering step, we will first run other perplexity parameter values to gauge how good Adjutant’s default choices are (similar to what was done in Wattenberg et.al.)

```r
df<-c()
for(tsnePer in c(2,5,30,50,100)){
  tsneObj<-Rtsne(sampleDat[,c("x","y")],perplexity = tsnePer)
  df<-rbind(df,cbind(1:nrow(tsneObj$Y),rep(tsnePer,nrow(tsneObj$Y)),tsneObj$Y))
}
df<-data.frame(df,stringsAsFactors = FALSE)
colnames(df)<-c("PMID","perplexity","tsneComp1","tsneComp2")

#let's take a look at the t-SNE output
df<-inner_join(sampleDat,df)

ggplot(df,aes(x=tsneComp1,y=tsneComp2,color=grp))+
  facet_grid(.~perplexity)+
  geom_point(alpha=0.7)+
  theme_bw()+
  labs(title="RTsne with variable perplexity parameters",
       subtitle="Theta = 0.5, trading off accuracy for more speed")
```
The figure above shows that the greater the perplexity parameter the “closer” we get to resolving two clusters – this makes sense since a largely perplexity parameters consults more neighbors prior in the dimensionality reduction process. This was also shown in Wattenberg et al. As the perplexity parameters get larger it is possible to see that while the exact spatial orientation and density is not the same the original, it is clear that are two slightly overlapping clusters. What is notable is that Adjutant’s t-SNE implementation is not performing the fully accurate t-SNE, since the rTsne package has a theta parameter that speeds up t-SNE computation at the cost of some accuracy. The default value of theta is 0.5, however, we can use a theta value of 0.0 to get the classical version of t-SNE:

```r
# Let's take a look at the t-SNE output
df<-inner_join(sampleDat,df)
```

```r
# join, by = "PMID"
```

```r
ggplot(df,aes(x=tsneComp1,y=tsneComp2,color=grp))+
facet_grid(.~perplexity,scales="free")+
geom_point(alpha=0.7)+
theme_bw()+
```
Although this figure is slightly different than the previous, the results are the same: with a greater perplexity value there is better discernibility between the two clusters (note that above we’ve freed-up the x-axis scales to make it easier to see that last group).

### 2.2.1.2 Running Adjutant’s hdbscan procedure

The big question is whether hdbscan (Adjutant’s clustering algorithm) can reliably cluster on the dimensionally reduced data. The hdbscan algorithm requires that the user specifies the minimum number of points (minPts) in a cluster. It can be difficult to decide the best minPts parameter value and so Adjutant automatically tries several different cluster sizes and selects the best one based upon the procedure specified in the implementation details.

```r
# Under Adjutant, a tsne perplexity parameter of 50 would have been selected for the data.
#now we can see all the choices adjutant cycles through
# we can see all the choices adjutant cycles through
tsneObj<-Rtsne(sampleDat[,c("x","y")],perplexity = 50)
df<-data.frame(PMID = 1:nrow(tsnemObj$Y),
                tsneComp1 = tsnemObj$Y[,1],
                tsneComp2 = tsnemObj$Y[,2])
# now we can run the optimal params method, which runs HDBSCAN and picks the best parameters
optOut<-optimalParam(df)
plists<-lapply(optOut$altChoices,function(x){
x$fitPlot +
  labs(title = paste("minPts=",x$minPt)) +
  theme(legend.position="none",
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank())})

cowplot::plot_grid(plotlist = plists, nrow = 2)
```
The above small multiple figures show how the data would be clustered using different hdbscan parameters. We can see that in spite of those the presence of what visually looks like smaller clusters, hdbscan finds two clusters in the data over several minPts values; the only exception is when minPts = 10 and three clusters are detected. Thus, we have demonstrated that even though t-SNE does in fact distort spatial positions and point densities, the hdbscan algorithm can confidently determine that there are indeed two distinct clusters. Among the different minPts options Adjutant will pick what it thinks is the a good choice, the procedure and results are shown below:

```r
sampleDat$PMID<-factor(sampleDat$PMID)
df$PMID<-factor(df$PMID)

tmp<-inner_join(sampleDat,optOut$retItems)

#add a group for noise
df<-df %>% mutate(grpRev = ifelse(tsneCluster == "0","Noise",paste(grp)))

#get cluster of co-ordinates
clusterNames <- df %>%
  dplyr::group_by(grpRev) %>%
  dplyr::summarise(medX = median(tsneComp1),
                   medY = median(tsneComp2)) %>%
  dplyr::filter(grpRev != "Noise")

#plotting the differences
resolved<-ggplot(df,aes(x=tsneComp1,y=tsneComp2,group=grpRev))+
  geom_point(alpha=0.7,aes(colour=grpRev))+
  scale_colour_manual(values=c(scales::hue_pal()(2),"lightgray"))+
  stat_ellipse(aes(alpha = grpRev))+
  geom_label(data=clusterNames,aes(x=medX,y=medY,label=grpRev),size=2,colour="black")+
```

Adjutant Supplemental Materials (Crisan et al.)
scale_alpha_manual(values = c(0.7,0.7,0.0))+
theme_bw()+
labs(title="Resolved Plot",
     subtitle="Data after t-SNE and HDBSCAN.
Ellipses denote HDBSCAN clusters on t-SNE data")

original<-ggplot(df,aes(x=x,y=y,color=grp))+
geom_point(alpha=0.7)+
theme_bw()+
labs(title="Original Plot",
     subtitle="Data before t-SNE and HDBSCAN")
cowplot::plot_grid(original,resolved,nrow=1)

The above figure shows the original data (left) as well as the resolved clusters once they’ve been run through Adjutant’s procedures (right). A couple of things are notable in the above figure. First, Adjutant correctly suggests that there are two clusters in the data. The spatial orientation and the point density of clusters are not identical to the original, but the structure of groups holds (i.e. points from grp 1 continue to cluster together after being run through Adjutant’s procedures). The next thing to note is that not all points could be classified. The hdbscan algorithm allows some points to be labelled as noise (not classified to any distribution, indicated in grey). When the ground truth is known this is not a desirable outcome, however, in situations where we are less certain about what the shape and structure of the data should be we argue this is a useful feature. There is no doubt that the two distributions overlap, and that there are some points at the peripheries of both distributions to which we would attach greater uncertainty. What Adjutant does successfully cluster has a very strong signal. If unclassified points are problematic, we suggest two ways to address the issue of noise. One approach is to force Adjutant to cluster as many of the points as possible and choose the minPts value that does just that – there are downsides to this approach since sometimes ambiguity or uncertainty is warranted when assigning a
point to a cluster. Another approach is to use the more confidently clustered data as a prior for other techniques that can classify items relegated to noise.

It is possible to conclude from this analysis that while t-SNE does misrepresent the original spatial positions of points it is still possible to find the correct number of clusters in the data. Adjutant’s goal is to suggest clusters and in this task, its algorithmic procedure does a good job. We do advise caution in overly interpreting spatial positions of clusters, although we will show soon spatial positions are not entirely irrelevant either.

2.2.2 A Multi-class Example

Instead of using two distributions, we’ll now simulate a more complex scenario. We use exactly the same procedures as were used in the simpler two-class example.

2.2.2.1 Generating sample data

```r
sampleDat <- c()
count = 1
prevPoint <- c()
while(count < 10){
  n = sample(30:250,1)
mu <- c(sample(-20:20,1),sample(-20:20,1))
sC <- sort(sample(0:5,4,replace=TRUE),decreasing = T)
Sigma <- matrix(c(sC[1],sC[3],sC[4],sC[2]),2,2)
tmp <- rbind(prevPoint,mu)
  # make sure that distributions are not sitting RIGHT on top of each other.
  if(sum(dist(tmp)<10)<1){
    prevPoint <- rbind(prevPoint,mu)
    sampleDat <- rbind(sampleDat,
      multiNormGen(n=n,mu=mu,Sigma = Sigma,grpName = paste("grp",count)))
    count = count + 1
  }
}
#Give the rows some IDS
sampleDat$PMID <- 1:nrow(sampleDat)
clusterNames <- sampleDat %>%
  dplyr::group_by(grp) %>%
  dplyr::summarise(medX = median(x),
    medY = median(y))
ggplot(sampleDat,aes(x=x,y=y,colour=grp,group=grp))+
  geom_point(alpha=0.7)+
  geom_label(data=clusterNames,aes(x=medX,y=medY,label=grp),size=8,colour="black")+
  stat_ellipse()+ scale_colour_manual(values=c(tableau_color_pal("tableau10medium")(10),"black"))+
  theme_bw()
```
Above is the are the new clusters. Note that they have different sizes and shapes, which Adjutant must resolve.

2.2.2.2 Running t-SNE

#t-SNE resolution at different perplexity parameters

def<-c()
for(tsnePer in c(2,5,30,50,100)){
    tsneObj<-Rtsne(sampleDat[,c("x","y")],perplexity = tsnePer)
    df<-rbind(df,cbind(1:nrow(tsneObj$Y),rep(tsnePer,nrow(tsneObj$Y)),tsneObj$Y))
}

df<-data.frame(df,stringsAsFactors = FALSE)
colnames(df)<-c("PMID","perplexity","tsneComp1","tsneComp2")

#Let's take a look at the t-SNE output

def<-inner_join(sampleDat,df)

ggplot(df,aes(x=tsneComp1,y=tsneComp2,color=grp))+
    facet_grid(.~perplexity,scales="free")+
    geom_point(alpha = 0.8)+
    theme_bw()+
    scale_colour_manual(values=c(tableau_color_pal("tableau10medium")(10),"black"))
As with the simpler two distributions, the higher the perplexity parameter the better t-SNE performs at resolving the individual clusters. Even a t-SNE parameter of just five starts to separate the data points.

### 2.2.2.3 Running Adjutant’s hdbscan Procedure

Following t-SNE, we will run the hdbscan procedure and show the clustering results with multiple minPts parameters that Adjutant automatically explores.

```r
# Under Adjutant, a tsne perplexity parameter of 50 would have been selected for the data.
# We can see all the choices adjutant cycles through

tsneObj <- Rtsne(sampleDat[,c("x","y")], perplexity = 50)
df <- data.frame(PMID = 1:nrow(tsneObj$Y),
                 tsneComp1 = tsneObj$Y[,1],
                 tsneComp2 = tsneObj$Y[,2])

# now we can run the optimal params method, which runs HDBSCAN and picks the best parameters
optOut <- optimalParam(df)

# we can see all the choices adjutant cycles through
pList <- lapply(optOut$altChoices, function(x){
  x$fitPlot +
  scale_colour_manual(values = c(tableau_color_pal("tableau10medium")[10], "black")) +
  labs(title = paste("minPts = " , x$minPt)) +
  theme(legend.position = "none",
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank())
})
cowplot::plot_grid(plotlist = pList, nrow = 2)
```
The above figure shows the clusters suggested by hbdscan using various minPts parameters. Between a minPts of 10 and 50, it is possible to resolve every single cluster. As the minPts parameter becomes greater than 75 visually separable groups become clustered together by the algorithm (i.e cluster 2 in the minPts = 75 panel) and some clusters with too few points are unclustered. Next, we will use Adjutant’s automatic minPts procedure to determine the final cluster structure. Again, Adjutant’s unsupervised clustering procedures has no “awareness” of the actual number of clusters. It will examine the distribution of clusters and make its best assessment.

```r
# some data prep
sampleDat$PMID <- factor(sampleDat$PMID)
df$PMID <- factor(df$PMID)
sampleDat <- inner_join(sampleDat, optOut$retItems)
df <- inner_join(df, sampleDat)

# add a group for noise
df <- df %>% mutate(grpRev = ifelse(tsneCluster == "0", "Noise", paste("hdbscan clust", tsneCluster)))

# get cluster of co-ordinates
clusterNames <- df %>%
  dplyr::group_by(grpRev) %>%
  dplyr::summarise(medX = median(tsneComp1),
                   medY = median(tsneComp2)) %>%
  dplyr::filter(grpRev != "Noise")

df <- df %>%
  mutate(isNoise = ifelse(grp == "Noise", "Noise", "Signal"))
```
# plot differences
```r
resolved <- ggplot(df, aes(x=tsneComp1, y=tsneComp2, group=grpRev)) +
  geom_point(aes(colour=grp, alpha = isNoise)) +
  stat_ellipse() +
  theme_bw() +
  scale_alpha_manual(values = c(0.2,0.7)) +
  scale_colour_manual(values=c(tableau_color_pal("tableau10medium")(10),"black")) +
  labs(title="Resolved Plot",
       subtitle="Data after t-SNE and HDBSCAN. Ellipses denote HDBSCAN clusters on t-SNE data")
```

```r
original <- ggplot(df, aes(x=x, y=y, color=grp)) +
  geom_point(aes(alpha = isNoise)) +
  theme_bw() +
  scale_alpha_manual(values = c(0.2,0.7)) +
  scale_colour_manual(values=c(tableau_color_pal("tableau10medium")(10),"black")) +
  labs(title="Original Plot",
       subtitle="Data before t-SNE and HDBSCAN")
```

cowplot::plot_grid(original, resolved, nrow=1)

Comparing the original plot against the t-SNE cluster plot a few things are noteworthy. Again, the t-SNE co-ordinates do not recover the identical spatial positions (and by extension point density) of the original plot, but it is clear that Adjutant’s procedures are none-the-less able to recover the original number of clusters even in this more complex example – which again is the goal of this tool. It turns out that in this example when the clusters are actually separable and not overlapping that none of the points are classified as noise.
This example also shows that the t-SNE positions are not irrelevant. For example, the red, brown, and blue groups are close to each other like in the original, as are the gray, purple, and orange clusters. The proximity of the blue cluster to the orange cluster is misleading, but this is likely an artifact of preserving its proximity to other clusters. As noted in Wattenberg et al., it is important to be cautious when interpreting spatial positions from t-SNE, but we will show below that even in unsupervised clusters of more complex document data sets there are ways to assess the validity of spatial positioning of points.

2.2.2.4 Adding Noise

Next, we’ll test the limits of Adjutant’s algorithms to see how well it does with some messier, but still synthetic, data, by adding about 1000 data points of noise.

```r
sampleDat <- dplyr::select(sampleDat, x, y, grp)

# Let's add some noise in the middle to keep things interesting
# Generally, I have found that noisy articles hang out in the middle,
# and really clearly resolvable clusters out to the sides.
sampleDat <- rbind(sampleDat,
                    multiNormGen(n=1000, mu=c(0, 0), Sigma = matrix(c(60, 35, 30, 55), 2, 2), grpName = "Noise"))

sampleDat$PMID <- 1:nrow(sampleDat)
```

![Simulated Data with Noise](image)

isNoise
- Noise
- Signal
group
- grp 1
- grp 2
- grp 3
- grp 4
- grp 5
- grp 6
- grp 7
- grp 8
- grp 9
- Noise
When adding the noise, it becomes clear that several clusters caught within the noise data points are lost. Clusters on the periphery (yellow, green, purple, pink, and brown) do indeed still form their own clusters and appear to also pick up some points from the “noise” distribution, whereas the yellow, orange, and blue clusters that are totally overlapped by the noise all get clumped together. If the noise points are diffuse enough it may be possible that dense pockets of points within it are clusterable, but this is not the case here.

2.3 Evaluation with Real Data

Relevant and real-world evaluative datasets for text mining are difficult to obtain. This is because so-called ‘ground truth data’ is still subjectively classified by some individual and a different individual, with different analysis goals, could classify the same dataset differently. Thus, absent reliable ground truth data, we sought to investigate whether Adjutant’s classification of articles into specific clusters is reasonable or is random. We do this by evaluating the distribution of term frequencies between clusters. We begin with a simple example where we simply look up the distribution of common terms that make up each cluster. We follow-up with a more complex example that compares the ‘bag-of-words’ between different clusters in order to demonstrate that articles within a cluster are more similar to each other than to articles in other clusters. The prior results with simulated data show that if there are clusters to find Adjutant is able to detect them. However, we still advise users to interpret Adjutant’s results and not simply take them as ground truth.

2.3.1 Obtaining Real-world Data and Running Adjutant

We will obtain a real-world dataset comprising of approximately 20,000 articles pertaining to the application of whole genome sequencing in medicine or public health. The main applications for whole genome sequencing tend to be human cancers or human pathogens, and it is reasonable to hypothesize that these two application contexts are sufficiently different such that they’d cluster into separate groups. We emphasize once more, that we do not have ground truth for these data.
We’ve used Adjutant’s workflow methods to obtain a document corpus and run unsupervised clustering procedures (t-SNE and hdbscan)

```r
df<-processSearch(('("whole genome" OR "next generation" OR "high throughput") AND "sequencing") AND ("medicine" or "public health"),retmax=20000)

tidy_df<-tidyCorpus(corpus = df)

#running t-SNE
    tsneObj<-runTSNE(tidy_df,check_duplicates=FALSE,perplexity=100)
    df<-inner_join(df,tsneObj$Y,by="PMID")

#running hdbscan
    optClusters <- optimalParam(df)
```

We can explore the ways the various hdbscan minPts parameter values would have clustered these data:

```r
pList<-lapply(optClusters$altChoices,function(x){
    x$fitPlot + 
    labs(title = paste("minPts=",x$minPt)) + 
    theme(legend.position="none", 
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank())
})

cowplot::plot_grid(plotlist = pList, nrow = 2)
```
The above figure shows that clusters forming toward the periphery of the t-SNE plot tend to be consistently grouped and that the changing minPts parameter appears to have the greatest effect on documents clustered toward the center of the plot (similar to what was shown with the noisy synthetic data analysis in Section 2.2.2.4). More specifically, as the minimum cluster size passed to hdbscan gets larger, we lose the ability to resolve clusters within the middle blob. Why are those clusters in the middle blob resolvable at all? It’s likely because the density within that blob is diffuse enough that smaller, but dense, clusters within it are still resolvable by hdbscan.

Finally, we can visualize the final set of clusters that Adjutant would settle upon after running the greedy algorithm to detect a good value for the minPts parameter value.

```r
#Merge best clusters with HBDSCAN
df<-inner_join(df,optClusters$retItems,by="PMID") 
  mutate(tsneClusterStatus = ifelse(tsneCluster == 0, "not-clustered","clustered"))

#now name the clusters
clustNames<-df %>%
  group_by(tsneCluster)%>
  mutate(tsneClusterNames = getTopTerms(clustPMID = PMID, clustValue=tsneCluster,top NVal = 2,tidyCorpus=tidy_df)) %>%
  select(PMID,tsneClusterNames) %>%
  ungroup()

#update document corpus with cluster names
df<-inner_join(df,clustNames,by=c("PMID","tsneCluster"))

#plot the clusters
clusterNames <- df %>%
```
Adjutant suggests 60 clusters for this document corpus of 20,000 articles. The above figure shows that there is some separation between cancer clusters (upper right-hand side) and public health clusters (lower left hand side). There are also documents pertaining to different molecular features (mutations, cnvs, snps) and sources of molecular data (protein, rna) etc which span between the cancer and public
health regions of the plot. We will next test whether the terms found within these clusters are specific enough to support the definition of these clusters.

2.3 Validity of clusters

To assess the validity of the clusters without ground truth data we look at the distribution of terms across t-SNE coordinates expecting that each cluster has terms that are specific to it and that can distinguish that cluster from those around it. Again, while precise spatial positions from t-SNE should be cautiously interpreted, we have shown with synthetic data in Section 2.2.2 that t-SNE spatial co-ordinates are not all together irrelevant. This is not a naïve assertion, as Wattenberg et. al. also shows the spatial positions, although different from the original, are still relevant - namely in the colored grid example the color gradient of the original is maintained after t-SNE was applied.

We use two methods to check the cluster validity. The first is a simple approach that considers the distribution of individual words and the second is a bit more complex that compares the similarity of ‘bags-of-words’ between clusters.

2.3.2 Visualizing the distribution of specific terms

In this example we will assess the distribution of terms that make up cluster names, since these reflect the top-two most common terms in the cluster itself. Ideally the top two terms of a cluster name should occur primarily within the cluster of interest, which is crude metric but a useful start. We’ll also include some common biology terms, specifically “cancer”, “gene”, “protein”, “expression”, “mutation”, and “study” which we hypothesize should occur across nearly all of the clusters.

We summarize the results are a figure of small multiples, were each individual chart shows the distributions of a term across all the clusters.

```r
# bag of terms
clustBag <- clusterNames %>%
  mutate(clusterWords = strsplit(tsneClusterNames, "-")) %>%
tidyr::unnest(clusterWords)

clustBag <- unique(c(clustBag$clusterWords, c("cancer","gene","data","protein","expression","mutat","studi"))

tmp <- df %>%
  filter(tsneClusterStatus == "clustered") %>%
dplyr::select(PMID,tsneComp1,tsneComp2,tsneClusterNames)

tidy_df_check %>%
  filter(wordStemmed %in% clustBag) %>%
  filter(tsneClusterStatus == "clustered") %>%
  ggplot(aes(x=tsneComp1,y=tsneComp2,group=tsneClusterNames))+
  stat_ellipse(data=tmp, aes(group=tsneClusterNames), col="red")+
  geom_point(alpha=0.2)+
  facet_wrap(~wordStemmed)+
  theme_bw()+
  theme(legend.position="none",
        panel.grid.major= element_blank(),panel.grid.minor = element_blank(),
```
In the above figure, each red circle represents the clusters Adjutant has derived, in each figure we show all of the clusters, but we’ve only drawn in the points for articles that contain a specific term. Although it is not very easy to read the individual terms, the pattern that is evident is that there are indeed some terms that tend to be present across all clusters in the dataset and others that are present only specific clusters within specific regions of the t-SNE plot. For example, “genom”, is a term that essentially present across all the clusters in the corpus, whereas terms like “outbreak”, “vaccine”, “lymphoma”, and even “tumor” tend to be present in specific clusters and areas of the plot. The clearest example of this separation is clusters containing the words “genom”, “strain”, and/or “tumor”:

```r
tidy_df_check %>%
  filter(wordStemmed %in% c("genom","strain","tumor")) %>%
  filter(tsnecClusterStatus == "clustered") %>%
  ggplot(aes(x=tsneComp1,y=tsneComp2,group=tsneClusterNames)) +
  stat_ellipse(data=tmp, aes(group=tsneClusterNames), col="red") +
  geom_point(alpha=0.2) +
  facet_wrap(~wordStemmed) +
  theme_bw() +
  theme(legend.position="none", panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  axis.text = element_blank(), axis.title = element_blank(),
```
The term “genom” occurs across all clusters, while “strain” tends to be present in clusters on the lower left quadrant and “tumor” tends to be present in clusters of the upper right quadrant. This suggests that there is some underlying “method to the madness” that is reasonable since there is some spatial relationship between the presence of specific terms and clusters, but this does not imply that the clusters are perfectly correct. This is still a very simple example and only based on single word matches. What is more useful, and also more complex, is to compare clusters as “bags of words”, and that is precisely what we’ll do now.

### 2.3.2.2 Visualizing cluster similarity

Showing the distribution of handful of terms gives us a sense that there are indeed some relevant merits of the spatial clustering on t-SNE, however, we’ll now implement a more sophisticated approach that treats each cluster as a bag of words (derived from its documents) and compares derived term frequencies between clusters using the cosine similarity. We expect that a cluster is most similar to its self and to some extent possible also similar to proximal clusters, but has little to no similarity to more distal clusters.

```r
#quickly calculate the cosine similarity
coss <- function(x) {crossprod(x)/(sqrt(tcrossprod(colSums(x^2))))}

#looking at term frequency across clusters
#note that t-SNE is based on the tf_idf metric
#instead, we're comparing term frequencies across documents
clustDTM<-tidy_df_check %>%
  select(-tf,-idf,-tf_idf) %>% #remove tf,idr,tf_idf from document level, now turn to cluster level
  group_by(tsneClusterNames,wordStemmed) %>%
  summarise(count = sum(n)) %>%
  ungroup() %>%
  bind_tf_idf(wordStemmed, tsneClusterNames, count) %>%
```
cast_dtm(.,tsneClusterNames,wordStemmed,tf) %>%
  as.matrix()

simClust<-reshape::melt(coss(t(clustDTM))) %>% data.frame()

colnames(simClust)<-c("tsneClusterNames","compClust","cosSim")

#draw how similar clusters are to other clusters around them based upon
#the terms that occur within them
pList<-c()
for(clust in unique(as.character(simClust$tsneClusterNames))){
  tmp<-filter(simClust,compClust == clust) %>% select(-compClust)
  tmp<-filter(tmp,tsneClusterNames != "Noise") %>%
    mutate(isCluster = ifelse(tsneClusterNames == clust,1,0)) %>%
    mutate(cosSimStep=ifelse(cosSim <= 0.2,0,1)) %>%
    ggplot(aes(x=tsneComp1,y=tsneComp2,group=tsneClusterNames,
               colour = factor(isCluster),fill = cosSimStep)) +
    stat_ellipse(geom="polygon") +
    scale_fill_manual(values=c("black","red"),name="Selected Cluster",labels=c("other",clust)) +
    labs(title=clust)+
    theme_bw()+
    theme(legend.position="none",
          panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
          axis.text = element_blank(),axis.title = element_blank(),
          panel.border = element_blank(),axis.ticks = element_blank(),
          strip.text = element_text(size=8))

  pList[[clust]]<-p
}

#Just looking at one for sanity
pList[[1]] + theme(legend.position = "right")
Before considering the results of all possible clusters, we’ll examine the results of just one to understand what is being shown. The above figure is the result for the cluster “cancer-breast” (circled in red) with all other derived clusters in this data. The comparison is accomplished by visually overlaying the cosine similarity of all clusters relative to the “cancer-breast” cancer. The darker the blue the more similar the clusters; a cosine similarity of one (dark blue) indicates that identical clusters, while a cosine similarity of 0 (white) indicates that clusters are totally different.

The above figure of the “cancer-breast” cluster shows what we’d hoped for. First our cluster of interest (“cancer-breast” outlined in red) is, appropriately, a dark blue color (cosine similarity of 1). The clusters that are closest to it are not as dark blue but appear to have a cosine similarity between 0.6 and 0.8 (related but not identical), and as we move further and further away from the “cancer-breast” cluster we see that the cosine similarity drops. This suggests that there is indeed some sanity both to spatial positions of the t-SNE clustering and also how hdbscan and Adjutant procedures have chosen the cluster boundaries.

Let’s look at all the other clusters now, to examine if the same trend occurs.
Looking at all of the clusters essentially recapitulates what’s in the previous single cluster figure. This means that t-SNE spatial positions are reasonable and driven by common terms in documents and are not arbitrarily placed. It’s also interesting to consider the “Noise” data (row 4, last column), which has high cosine similarity with pretty much every cluster - this is another nice sanity check as these articles had a difficult time clustering within any particular cluster. Those unclustered documents could perhaps belong to two clusters, or were on the peripheries of established clusters, which yields the results that we see. Advanced users may use Adjutant’s derived clusters to classify those unclustered documents to discover whether or not they belong to one or more clusters. Adjutant doesn’t support that kind of further article classification because it is up to a user what they’d like to do next with their analysis. However, Adjutant’s R-compatible outputs allow the user to leverage the full complement of R’s analytic tools to further explore this document corpus from a more informed initial point.

3.0 Comparison to LDA

Topic modeling is a broad area and has had various advances in a number of different disciplines. Here we’ll compare against a standard in R, which is the LDA analysis presented in the tidytext manual. While many alternative approaches exist in R and beyond, we feel that is this is the most relevant comparison since we are developing an R-based tool and use tidytext’s features within Adjutant’s development.

One challenge of LDA, and many clustering methods, is that it’s not very easy to establish the right initial parameters that should be provided to the method. Adjutant tries to scan for a good parameters,
but the best way to do this remains an active area of research. The primary Tidytext manual example begins with *a priori* knowledge that there are four books, and thus initializes the K parameter (number of clusters) to 4. But in real world data, we don’t know how many clusters we should have because we’re taking a purely unsupervised approach. However, we do know that Adjutant would suggest around 60 clusters for this document corpus, so we can begin LDA with that.

We follow the same analysis as in the Tidytext manual. Input to LDA will be a document term matrix using and an initial cluster size of 60. We will then look at the top 5 most common terms within each cluster, shown below; the $\beta$ values from the LDA analysis provides a sense of how important a word is to a particular topic cluster.

dtm <- tidy_df %>%
    tidytext::cast_dtm(PMID, wordStemmed, tf_idf)

# suggested # of topics from Adjutant
adjClust <- length(unique(df$tsneClusterNames)) # 60

# run LDA
df_lda <- LDA(dtm, k = 60, control = list(seed = 1234))

# Extracting topics
df_topics <- tidy(df_lda, matrix = "beta")

# top words within each topic
top_terms <- df_topics %>%
    group_by(topic) %>%
    top_n(5, beta) %>%
    ungroup() %>%
    arrange(topic, -beta)

# let's look at this as a tile plot
ggplot(top_terms, aes(x = topic, y = term, fill = beta)) +
    geom_tile() +
    theme_bw()
The above figure attempts to show the top five most value terms for each cluster and has colored each point according to the values of $\beta$ for that term-cluster combination. It is evident that some words belong to multiple topics – this is not surprising as we’ve already seen that terms like “genom” can be readily found across many documents. However, the $\beta$ values are all dark blue (closer to zero), implying that there is not a strong association between some particular term and a topic cluster. This is different than what the t-SNE analysis revealed, which showed that there indeed some individuals words that could be quite prominent in some clusters, or spatially proximal clusters, and absent or less present in other clusters.

It is also possible to analyze the topics that individual articles belong to by using the $\gamma$ values from the LDA. The values from the $\gamma$ distribution will vary between 0 and 1, and ideally, we would like to see values closer to 1 (documents were confidentially assigned to some topic).

```
df_gamma<- tidy(df_lda, matrix = "gamma")

#examining the gamma distributions
```
The above figure suggests that it’s hard for LDA to place documents within topics since the γ values are between 0 and 0.08 (very low).

It is possible to try a range of different values to assess whether LDA could improve the clustering results. However, with 20,000 documents, there is a huge range of potential values that can be passed to LDA’s ‘k’ parameter function. It quickly becomes cumbersome and slow to try all of these different parameters. One large difference between LDA and Adjutant’s unsupervised clustering implementation is that Adjutant does not need to classify every document, but LDA tries to do so. This difference may be why Adjutant finds distinct clusters in this data and LDA does not.

We can also see that the combination of t-SNE and hbscan allowed us to reason about the topic clusters a little but more easily, since they can be visually inspected (as we showed earlier) and since “minimum cluster size” is a far easier parameter to reason about than the expected “number of clusters”. It may be to possible to improve the LDA results by trying different parameter combinations, cleaning the data in different ways, or using other packages. But herein lies the problem Adjutant is trying to address. The combination of t-SNE and hbscan does a reasonably good first pass, which can help an individual to reason about the next steps from a more informed position.